Learning finite functions by neural networks : Evaluation of Pentago positions by convolutional neural networks (Algebras, logics, languages and related areas)

AUTHOR(S):

Jimbo, Shuji

CITATION:

Jimbo, Shuji. Learning finite functions by neural networks : Evaluation of Pentago positions by convolutional neural networks (Algebras, logics, languages and related areas). 数理解析研究所講究録 2018, 2096: 25-31

ISSUE DATE:

2018-12

URL:

http://hdl.handle.net/2433/251730

数理解析研究所講究録 第2096巻 2018年 25-31

Learning finite functions by neural networks: Evaluation of Pentago positions by convolutional neural networks

Shuji JIMBO
Graduate School of Natural Science and Technology, Okayama University

jimbo-s@okayama-u.ac.jp

Abstract

A convolution neural network (CNN) is a useful tool that approximates a finite function. It is used as a solver for various problems in the real world. In this paper, results of experiments on training variations of a small CNN used for image recognition for evaluating Pentago positions are mainly reported. The author hopes that the results are used in discussion of applicability of deep neural networks to researches in theoretical computer science.

KEYWORDS. convolutional neural network, computer experiment, two-player abstract strategy game, primality test.

1 Introduction

Convolutional neural networks (CNN's) are widely used in pattern- and image-recognition problems. Each neuron in convolutional layers connects with neurons in a restricted small area in the previous layer. CNN's are also much effective for evaluation of game positions. AlphaGo, the AI Go player developed by Google DeepMind, is a most famous example for the purpose[4][5]. Evaluation of positions of two-player abstract strategy games like Go is theoretically possible by minimax tree search of the complete game tree. However, it is impossible in practice. For this reason, tree search of a shallow sub-tree around the root of the game tree is used along with approximate evaluation of positions in practice. The strength of AlphaGo demonstrates the usefulness of deep neural networks for finding good moves in two-player abstract strategy games.

In this paper, we will report the results of experiments on application of small CNN's for image recognition to uses other than image recognition. They are evaluation of Pentago positions, acquisition of a simple rule based on a particular area, and primality testing. Pentago is a small two-player abstract strategy game. We will chiefly express experiments on evaluation of Pentago positions by deep CNN's.

The purpose of this paper is to provide results of experiments on the ability of neural networks to learn finite functions. We hope that those results will be used to consider the applicability of neural networks in theoretical computer science. For example, we might be able to efficiently search a graph that has a desirable condition with help of deep neural networks.

2 Board game Pentago

Pentago is a small two-player abstract strategy game invented by Tomas Flodén. It has been strongly solved by Geoffrey Irving with a supercomputer, NERSC's Cray Edison[3]. A part of the data obtained by retrograde analyses is released in the public domain. The size of the data is about four terabytes.

The board of Pentago is of 6×6, and it is divided into four 3×3 sub-boards or quadrants. The following describe how to play Pentago: Beginning with an empty board, two players alternately put a stone of the player's color on the board and then rotate a quadrant by 90 degrees either clockwise or anticlockwise; A pass is not allowed under any circumstances. The rules of victory or defeat is as follows: If only one player gets five stones of the player's color in a row, either horizontally, vertically, or diagonally, then the game is the player's win; If both players get five in a row at the same time, or the last move is played with no five in a row, then the game is a draw.

Figure 1 shows a progress of Pentago. Board A is black's turn. By putting a black stone (Board B), and then by rotating a quadrant (Board C), black can win.

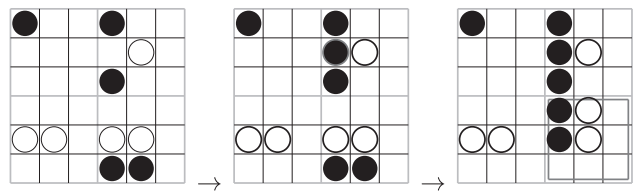

Figure 1: A progress of Pentago.

A player will get an advantage if the player has put a stone on the board. Black therefore can simulate the white's strategy more advantageously than white. The following proposition is clearly hold.

Proposition 1 *The second player, i.e. white, cannot win with perfect play.*

We show the scale of Pentago. The numbers of positions of slice n in the data are shown in Table 1. Those numbers are obtained by counting the positions contained in the data released in the public domain, including a lot of duplicate positions.

Table 1: The scale of Pentago.

0	256	7	144645120	14	2180723700736
1	768	8	832507904	15	5720741273600
2	9216	9	3550828544	16	14930328007168
3	73216	10	15994707968	17	31389244375296
4	720896	11	60293679104	18	65007135675648
5	4037632	12	231269590016		
6	27040768	13	708482302976		

In this research, only slices 11 and 12 positions were used due to restrictions of our computing environment. The numbers of positions in which black is to be win, white is

to be win, and the game is to be drawn are shown in Table 1. Let C denote a player black or white, and $\overline{C}$ the opposite player to C. The ratio of wins of C at C's turn is larger than that at $\overline{C}$'s turn. And, the ratio of wins of black at black's turn is very large.

Table 2: The details of positions in slice 11 and 12.

	slice 11	slice 12
Draw	15116993859	20753197343
Black win	14013289093	199909936103
White win	31163396152	10606456570

The data set used in computer experiments was made as follows. First, sufficiently large number of positions were randomly selected according to uniform distribution. Then, selected positions were thinned out so that the ratio among draw, black win, and white win positions is almost equal. The size of mini batch is 128. The number of training data is 850,000 batches (108,800,000 samples) for both slices. The number of samples in test data is just 1,000,000.

3 Convolutional Neural Networks

Neural networks can be tools to approximate finite functions, which are functions whose domain is a finite set. The result with perfect play from a given position of Pentago is an example of a finite function of middle difficulty for learning by neural networks. For learning by neural networks, decryption is a representative of the hardest finite functions. Whereas acquisition of a simple rule based on a particular area in a input image is regarded as a very easy finite function.

A CNN model for CIFAR-10 classification is released as an demonstration of a TensorFlow tutorial. We call the CNN the TensorFlow tutorial CNN for CIFAR-10. CIFAR-10 classification is a common benchmark problem to classify RGB 32×32 pixel images across 10 categories.

Evaluation of Pentago positions can be regarded as a problem to classify three color 6×6 pixel images across three categories: black win, draw, and white win. We attempted experiments on evaluation of Pentago positions by modified versions of the TensorFlow tutorial CNN for CIFAR-10. Then, we obtain the following results: The ratio of correct answers is about 85 percent for slice-12 positions, and about 75 percent for slice-11 positions, where a slice-n position of Pentago is a board on which n stones are placed.

The structures of TensorFlow tutorial CNN's for CIFAR-10 is described in a TensorFlow tutorial website in detail[1]. Basic structure of the CNN's used for our experiments is described in Figure 2. Pooling and normalization layers are removed in our CNN's so that our CNN's are sensitive to stone distribution.

Variations in structure of our CNN's used in computer experiments are as follows:

CNN A Basic CNN.

CNN B CNN obtained by increasing the number of filters of basic CNN from 64 to 128.

CNN C CNN obtained by increasing the number of convolutional layers of basic CNN from 2 to 4.

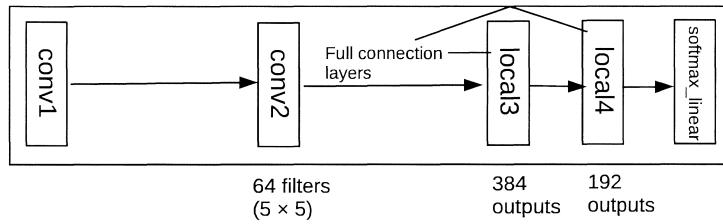

Figure 2: Basic structure of our CNN's.

Table 3: Ability of the CNN's for evaluating Pentago positions.

	X_{11}	Y_{11}	X_{12}	Y_{12}
CNN A	0.810	0.477	0.842	0.389
CNN B	0.814	0.459	0.846	0.398
CNN C	0.809	0.442	0.838	0.393
CNN D	0.808	0.463	0.842	0.390
CNN E	0.779	0.592	0.818	0.499
CNN F	0.779	0.577	0.813	0.518
CNN G	0.579	0.932	0.665	0.828

CNN D CNN obtained by both modifications in CNN B and CNN C.

CNN E CNN obtained by reducing the number of filters of basic CNN from 64 to 32.

CNN F CNN obtained by reducing the number of convolutional layers of basic CNN from 2 to 1.

CNN G CNN obtained by removing all of the convolutional layers from basic CNN.

Initial weights of those CNN's are small values randomly selected according to uniform distribution.

Table 3 shows ability of the CNN's for evaluating Pentago positions obtained by machine learning, where X_i and Y_i denote the accuracy rate measured with test data and the cross entropy error of the last 1,000 batches of training data for slice i positions respectively, where i is 11 or 12.

Table 3 shows that even if the scale of a CNN is increased from the basic CNN, there is almost no change in the accuracy rate and the cross entropy error. Furthermore, it also shows that if the scale is reduced, the accuracy rate decreases and the cross entropy error increases accordingly.

We currently think that the reason why the accuracy rate of evaluating Pentago positions by our CNN's is less than 87 percent is because the following conjecture holds. For each $n \in \{11, 12\}$, there is a set of Pentago positions of slice n that simple CNN's can hardly recognize, and the size of the set is more than ten percent of whole slice n positions. We call the set of Pentago positions in the conjecture above the hard core of the slice n positions.

An output of a CNN that classifies Pentago positions across three categories is called a logit, a 3-dimensional vector, such that each component is nonnegative and the sum of

three components is 1. The answer from such a CNN is the category corresponding to the maximum component. Hence, credibility of the answer from a CNN can be estimated based on its logit.

We performed the following experiment.

(a) First, 80 million slice-12 black win positions, 80 million slice-12 white win positions, and 80 million slice-12 draw positions were randomly selected according to uniform distribution. Then, data of 240 million slice-12 positions were made by shuffling the union of the three sets of 80 million positions. Then, the last 1,536,000 samples were used as test data α, and the other samples were used as training data α. Training CNN A with test data α resulted in CNN α.

(b) There are 10,220,000 samples x in training data α that satisfies the following condition. Training data β consists of all those samples.

> Condition: Let (a, b, c) be the logit corresponding to x. Assume that $\{v_1, v_2, v_3\} = \{a, b, c\}$, where the both sides are regarded as multisets. Furthermore, assume that $v_1 \geqq v_2 \geqq v_3$. Then, $(v_1 - v_2)/v_1 \leqq 0.6$ holds.

There are 313,741 samples x in test data α that satisfies the condition above. Test data β consists of all those samples.

(c) Training CNN A with test data β resulted in CNN β.

(d1) The accuracy rate of CNN α to test data α is 0.845.

(d2) The accuracy rate of CNN α to test data β is 0.575.

(d3) The accuracy rate of CNN β to test data α is 0.846.

(d4) The accuracy rate of CNN β to test data β is 0.590.

The results of the experiment above suggest a possibility that we can train a CNN efficiently and obtain a CNN with higher ability by using training data made from only hard core positions.

4 Application of TensorFlow tutorial CNN's for CIFAR-10 to other problems

We performed an experiment on ability of CNN's for acquisition of a simple rule based on a particular area.

Replacing the labels of the training data used in experiments on evaluation of Pentago positions with ones that stand for a particular figure. Then, the basic CNN was trained with those modified data. The accuracy rate achieved was 100 percent, and the cross entropy error of the last 1,000 batches of training data was 0.0. In other words, the trained CNN perfectly acquired a simple rule based on a particular area in a input image.

For each sample, the replaced label was determined by the numbers of black and white stones in the area enclosed by the bold boundary line in Figure 3 as follows: 1 if

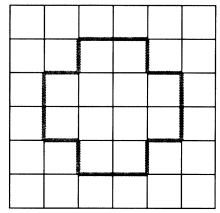

Figure 3: The area used in the definition of replaced labels.

black stones are more than white ones, 2 if white stones are more than black ones, and 0 otherwise.

Now, we also performed an experiment on ability of CNN's for testing primality.

The number of primes between 2^{17} and 2^{18} is 10,749. Let S_C denote the set of all products of two such primes including square numbers. The size of S_C is 57,775,875. Let S_P denote the set of randomly selected 57,775,875 prime numbers between 2^{35} and 2^{36}. Let T_C and T_P denote the sets of randomly selected 500,000 elements of S_C and S_P respectively.

The test data was made by shuffling $T_C \cup T_P$. The training data was made by shuffling $(S_C - T_C) \cup (S_P - T_P)$. Each sample is 36 bit binary integer written on a 6×6 matrix. The accuracy rate achieved was about 72 percent, and the cross entropy error of the last 1,000 batches of training data was about 0.5. The result shows that CNN's can acquire some degree of primality testing ability by training, confirming the high applicability of CNN's.

5 Concluding Remarks

We performed experiments on evaluation of Pentago positions of slice 11 or 12 by CNN's based on TensorFlow tutorial CNN's for CIFAR-10, and obtained the result that the accuracy rates of those CNN's are between about 80 and 85 percent. Furthermore, we performed an experiments on ability of CNN's for recognizing a particular figure, resulting in a CNN with perfect ability. And we performed an experiments on ability of CNN's for testing primality, resulting in a CNN of which accuracy rate achieved was about 72 percent. We hope that those results will be used to consider the applicability of neural networks to theoretical computer science.

We have two plans to increase dramatically the accuracy rate of evaluating Pentago positions with neural networks. First, we are planning to replace our CNN's with CNN's based on Deep Residual Networks or ResNets[2]. ResNets won the ImageNet and COCO competitions in 2015. Derivations of ResNets are widely used for various problems in the real world. Second, we are planning to form a combination of neural networks and game-tree search. Concerning AlphaGo Zero, the raw neural network, without using any look-ahead, achieved an Elo rating of 3,055, whereas the full system, that is the combination of the neural network and tree search, achieved a rating of 5,185[5].

Acknowledgment

We appreciate kindly discussion by the members in the RIMS workshop. This work was supported by JSPS KAKENHI Grant Number JP15K00018.

References

[1] Convolutional neural networks — tensorflow. `https://www.tensorflow.org/tutorials/deep_cnn`. Referred on 31 May 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Geoffrey Irving. Pentago is a first player win: Strongly solving a game using parallel in-core retrograde analysis. *arXiv preprint arXiv:1404.0743*, 2014.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[5] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.